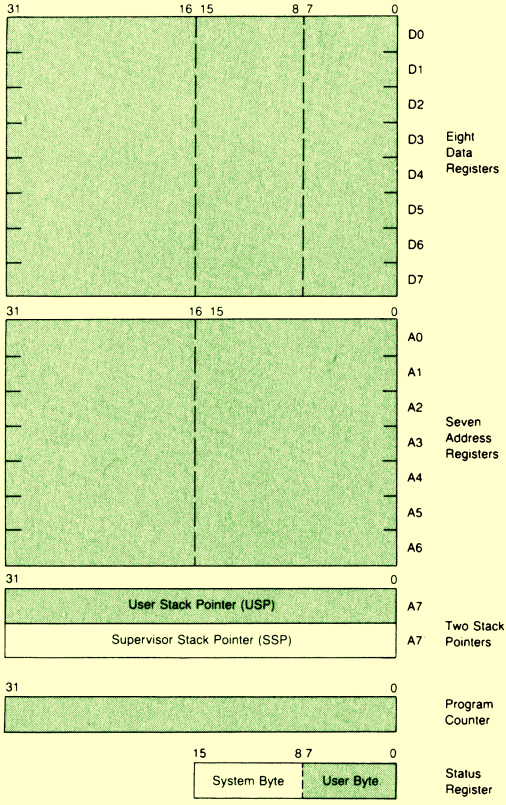
MOTOROLA

MC68008

8-/32-BIT MICROPROCESSOR (8-BIT DATA BUS) PROGRAMMING REFERENCE CARD

Programming Model



Condition Code Computations

Operations	X	N	Z	V	C	Special Definition
ABCD	*	U	?	U	?	C = Decimal Carry Z = Z • $\overline{Rm}$ • ... • $\overline{R0}$
ADD, ADDI, ADDQ	*	*	*	?	?	V = Sm • Dm • $\overline{Rm}$ + $\overline{Sm}$ • $\overline{Dm}$ • Rm C = Sm • Dm + $\overline{Rm}$ • Dm + Sm • $\overline{Rm}$
ADDX	*	*	?	?	?	V = Sm • Dm • $\overline{Rm}$ + $\overline{Sm}$ • $\overline{Dm}$ • Rm C = Sm • Dm + Rm • Dm + Sm • $\overline{Rm}$ Z = Z • $\overline{Rm}$ • ... • $\overline{R0}$
AND, ANDI, EOR, EORI, MOVEQ, MOVE, OR, ORI, CLR, EXT, NOT, TAS, TST	—	*	*	0	0	
CHK	—	*	U	U	U	
SUB, SUBI, SUBQ	*	*	*	?	?	V = $\overline{Sm}$ • Dm • $\overline{Rm}$ + Sm • $\overline{Dm}$ • Rm C = Sm • $\overline{Dm}$ + Rm • $\overline{Dm}$ + Sm • Rm
SUBX	*	*	?	?	?	V = $\overline{Sm}$ • Dm • $\overline{Rm}$ + Sm • $\overline{Dm}$ • Rm C = Sm • $\overline{Dm}$ + Rm • $\overline{Dm}$ + Sm • Rm Z = Z • $\overline{Rm}$ • ... • R0
CMP, CMPI, CMPM	—	*	*	?	?	V = $\overline{Sm}$ • Dm • $\overline{Rm}$ + Sm • $\overline{Dm}$ • Rm C = Sm • $\overline{Dm}$ + Rm • $\overline{Dm}$ + Sm • Rm
DIVS, DIVU	—	*	*	?	0	V = Division Overflow
MULS, MULU	—	*	*	0	0	
SBCD, NBCD	*	U	?	U	?	C = Decimal Borrow Z = Z • $\overline{Rm}$ • ... • $\overline{R0}$

Operations	X	N	Z	V	C	Special Definition
NEG	*	*	*	?	?	V = Dm • Rm, C = Dm + Rm
NEGX	*	*	?	?	?	V = Dm • Rm, C = Dm + Rm Z = Z • $\overline{Rm}$ • ... • $\overline{R0}$
BTST, BCHG, BSET, BCLR	—	—	?	—	—	Z = $\overline{Dn}$
ASL	*	*	*	?	?	V = Dm • ($\overline{D_{m-1}}$ + ... + $\overline{D_{m-r}}$) + $\overline{Dm}$ • (D_{m-1} + ... + D_{m-r}) C = D_{m-r+1}
ASL (r = 0)	—	*	*	0	0	
LSL, ROXL	*	*	*	0	?	C = D_{m-r+1}
LSR (r = 0)	—	*	*	0	0	
ROXL (r = 0)	—	*	*	0	?	C = X
ROL	—	*	*	0	?	C = D_{m-r+1}
ROL (r = 0)	—	*	*	0	0	
ASR, LSR, ROXR	*	*	*	0	?	C = D_{r-1}
ASR, LSR (r = 0)	—	*	*	0	0	
ROXR (r = 0)	—	*	*	0	?	C = X
ROR	—	*	*	0	?	C = D_{r-1}
ROR (r = 0)	—	*	*	0	0	

NOTES:
- Sm Source Operated — most significant bit
- Dm Destination operand — most significant bit
- ? See Special Definition
- Rm Result operand — most significant bit
- n bit number
- r shift count

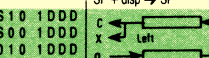
Effective Addressing Mode Categories

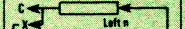
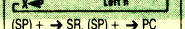
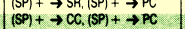
Type	Mode	Register	Generation	Assembler Syntex
Data Register Direct	000	reg. no.	EA = Dn	Dn
Address Register Direct	**001**	**reg. no.**	**EA = An**	**An**
Register Indirect	010	reg. no.	EA = (An)	(An)
Postincrement Register Indirect	**011**	**reg. no.**	**EA = (An), An←An + N**	**(An) +**
Predecrement Register Indirect	100	reg. no.	An←An − N, EA = (An)	− (An)
Register Indirect With Offset	**101**	**reg. no.**	**EA = (An) + d_{16}**	**d_{16}(An)**
Indexed Register Indirect With Offset	110	reg. no.	EA = (An) + (Xn) + d_8	d_8(An, Xn)
Absolute Short	**111**	**000**	**EA = (Next Word)**	**xxx**
Absolute Long	111	001	EA = (Next Two Words)	xxxxxx
PC Relative With Offset	**111**	**010**	**EA = (PC) + d_{16}**	**d_{16}(PC)**
PC Relative With Index and Offset	111	011	EA = (PC) + (Xn) + d_8	d_8(PC + Xn)
Immediate	**111**	**100**	**Data = Next Word(s)**	**#xxx**
Quick Immediate	—	—	Inherent Data	#xxx (1–8)
Implied Register	**—**	**—**	**EA = SR, USP, SP, PC**	**—**

NOTES:
- EA = Effective Address
- An = Address Register
- Dn = Data Register
- Xn = Address or Data Register used as Index Register
- SR = Status Register
- PC = Program Counter
- d_8 = Eight bit Offset (displacement)
- d_{16} = Sixteen bit Offset (displacement)
- N = 1 for Byte, 2 for Words and 4 for Long Words
- ( ) = Contents of
- ← = Replaces

Addressing Modes

Mnemonic	Size	Address Mode		Dn #	Dn ~	An #	An ~	(An) #	(An) ~	(An)+ #	(An)+ ~	−(An) #	−(An) ~	d_{16}(An) #	d_{16}(An) ~	dg(An,Xn) #	dg(An,Xn) ~	Abs.W #	Abs.W ~	Abs.L #	Abs.L ~	d_{16}(PC) #	d_{16}(PC) ~	dg(PC,Xn) #	dg(PC,Xn) ~	s = immed d = SR/CC #	s = immed d = SR/CC ~	Opcode Bit Pattern 1111 11 5432 1098 7654 3210	Boolean	Condition Codes X N Z V C
ABCD	B	s = Dn	d =	2	10																							1100 RRR1 0000 0rrr	d10 + s10 + X → d	* U * U *
		s = −(An)	d =									2	20															1100 RRR1 0000 1rrr		
ADD	**B**	**s = Dn**	**d =**	2	8		ADDA	2	16	2	16	2	18	4	24	4	26	4	24	6	32							1101 DDD1 00EE EEEE	d + Dn → d	* * * * *
		d = Dn	**s =**	2	8			2	12	2	12	2	14	4	20	4	22	4	20	6	28	4	20	4	22	4	16	1101 DDD0 00ee eeee	Dn + s → Dn	
	W	**s = Dn**	**d =**	2	8		ADDA	2	24	2	24	2	26	4	32	4	34	4	32	6	40							1101 DDD1 01EE EEEE	d + Dn → d	
		d = Dn	**s =**	2	8	2	8	2	16	2	16	2	18	4	24	4	26	4	24	6	32	4	24	4	26	4	16	1101 DDD0 01ee eeee	Dn + s → Dn	
	L	**s = Dn**	**d =**	2	12		ADDA	2	40	2	40	2	42	4	48	4	50	4	48	6	56							1101 DDD1 10EE EEEE	d + Dn → d	
		d = Dn	**s =**	2	12	2	12	2	26	2	26	2	28	4	34	4	36	4	34	6	42	4	34	4	36	6	28	1101 DDD0 10ee eeee	Dn + s → Dn	
ADDA	W	d = An	s =	2	12	2	12	2	20	2	20	2	22	4	28	4	30	4	28	6	36	4	28	4	30	4	20	1101 AAA0 11ee eeee	An + s → An	- - - - -
	L	d = An	s =	2	12	2	12	2	26	2	26	2	28	4	34	4	36	4	34	6	42	4	34	4	36	6	28	1101 AAA1 11ee eeee		
ADDI	**B**	**s = Imm**	**d =**	4	16		ADDA	4	24	4	24	4	26	6	32	6	34	6	32	8	40							0000 0110 00EE EEEE	d + # → d	* * * * *
	W	**s = Imm**	**d =**	4	16		ADDA	4	32	4	32	4	34	6	40	6	42	6	40	8	48							0000 0110 01EE EEEE		
	L	**s = Imm**	**d =**	6	28		ADDA	6	56	6	56	6	58	8	64	8	66	8	64	10	72							0000 0110 10EE EEEE		
ADDQ	B	s = Imm3	d =	2	8			2	16	2	16	2	18	4	24	4	26	4	24	6	32							0101 QQQ0 00EE EEEE	d + # → d	* * * * *
	W	s = Imm3	d =	2	8	2	8	2	24	2	24	2	26	4	32	4	34	4	32	6	40							0101 QQQ0 01EE EEEE		
	L	s = Imm3	d =	2	12	2	12	2	40	2	40	2	42	4	48	4	50	4	48	6	56							0101 QQQ0 10EE EEEE		
ADDX	**B**	**s = Dn**	**d =**	2	8																							1101 RRR1 0000 0rrr	d + s + X → d	* * * * *
		s = −(An)	**d =**									2	22															1101 RRR1 0000 1rrr		
	W	**s = Dn**	**d =**	2	8																							1101 RRR1 0100 0rrr		
		s = −(An)	**d =**									2	50															1101 RRR1 0100 1rrr		
	L	**s = Dn**	**d =**	2	12																							1101 RRR1 1000 0rrr		
		s = −(An)	**d =**									2	58															1101 RRR1 1000 1rrr		
AND	B	s = Dn	d =	2	8			2	16	2	16	2	18	4	24	4	26	4	24	6	32							1100 DDD1 00EE EEEE	d<and>Dn → d	- * * 0 0
		d = Dn	s =	2	8			2	12	2	12	2	14	4	20	4	22	4	20	6	28	4	20	4	22	4	16	1100 DDD0 00ee eeee	Dn<and>s → Dn	
	W	s = Dn	d =	2	8			2	24	2	24	2	26	4	32	4	34	4	32	6	40							1100 DDD1 01EE EEEE	d<and>Dn → d	
		d = Dn	s =	2	8			2	16	2	16	2	18	4	24	4	26	4	24	6	32	4	24	4	26	4	16	1100 DDD0 01ee eeee	Dn<and>s → Dn	
	L	s = Dn	d =	2	12			2	40	2	40	2	42	4	48	4	50	4	48	6	56							1100 DDD1 10EE EEEE	d<and>Dn → d	
		d = Dn	s =	2	12			2	26	2	26	2	28	4	34	4	36	4	34	6	42	4	34	4	36	6	28	1100 DDD0 10ee eeee	Dn<and>s → Dn	
ANDI	**B**	**s = Imm**	**d =**	4	16			4	24	4	24	4	26	6	32	6	34	6	32	8	40							0000 0010 00EE EEEE	d<and># → d	- * * 0 0
	W	**s = Imm**	**d =**	4	16			4	32	4	32	4	34	6	40	6	42	6	40	8	48							0000 0010 01EE EEEE		
	L	**s = Imm**	**d =**	6	28			6	56	6	56	6	58	8	64	8	66	8	64	10	72							0000 0010 10EE EEEE		
ANDI CCR	B	s = Imm	d =																							4	32	0000 0010 0011 1100	s<and>CCR → CCR	* * * * *
ANDI SR	**W**	**s = Imm**	**d =**																							4	32	0000 0010 0111 1100	s<and>SR → SR	* * * * *
ASL, ASR	B/W	count = Dn	d =	2	10 + 2n																							1110 rrrf SS10 0DDD	C ← 0	* * * * *
		count = #1–8	d =	2	10 + 2n																							1110 QQQf SS00 0DDD	X ← Left	
	L	count = Dn	d =	2	12 + 2n																							1110 rrrf 1010 0DDD	0 → C	
		count = #1–8	d =	2	12 + 2n																							1110 QQQf 1000 0DDD	X → Right	
	Memory	W	count = 1	d =					2*	24	2*	24	2*	26	4*	32	4*	34	4*	32	6*	40							1110 000f 11EE EEEE		
Bcc	**B**		**d_8 =**																					branch taken	2		18	0110 CCCC PPPP PPPP	If cc true, then PC + disp → PC	- - - - -
																								branch not taken	2		12			
	W		**d_{16} =**																					branch taken	4		18			
																								branch not taken	4		20			
BCHG	B	bit# = Dn	d =					2	16	2	16	2	18	4	24	4	26	4	24	6	32							0000 rrr1 01EE EEEE	~(bit#) of d → Z, ~(bit#) of d → (bit#) of d	- - * - -
		bit# = Imm	d =					4	24	4	24	4	26	6	32	6	34	6	32	8	40							0000 1000 01EE EEEE		
	L	bit# = Dn	d =	2	<12																							0000 rrr1 01EE EEEE		
		bit# = Imm	d =	4	<20																							0000 1000 01EE EEEE		
BCLR	**B**	**bit# = Dn**	**d =**					2	16	2	16	2	18	4	24	4	26	4	24	6	32							0000 rrr1 10EE EEEE	~(bit#) of d → Z, 0 → (bit#) of d	- - * - -
		bit# = Imm	**d =**					4	24	4	24	4	26	6	32	6	34	6	32	8	40							0000 1000 10EE EEEE		
	L	**bit# = Dn**	**d =**	2	<14																							0000 rrr1 10EE EEEE		
		bit# = Imm	**d =**	4	<22																							0000 1000 10EE EEEE		
BRA	B		d_8 =																							2	18	0110 0000 PPPP PPPP	PC + disp → PC	- - - - -
	W		d_{16} =																							4	18			
BSET	**B**	**bit# = Dn**	**d =**					2	16	2	16	2	18	4	24	4	26	4	24	6	32							0000 rrr1 11EE EEEE	~(bit#) of d → Z, 1 → (bit#) of d	- - * - -
		bit# = Imm	**d =**					4	24	4	24	4	26	6	32	6	34	6	32	8	40							0000 1000 11EE EEEE		
	L	**bit# = Dn**	**d =**	2	<12																							0000 rrr1 11EE EEEE		
		bit# = Imm	**d =**	4	<20																							0000 1000 11EE EEEE		
BSR	B		d_8 =																							2	34	0110 0001 PPPP PPPP	PC → −(SP), PC + disp → PC	- - - - -
	W		d_{16} =																							4	34			
BTST	**B**	**bit# = Dn**	**d =**					2	12	2	12	2	14	4	20	4	22	4	20	6	28	4	20	4	22			0000 rrr1 00EE EEEE	~(bit#) of d → Z	- - * - -
		bit# = Imm	**d =**					4	20	4	20	4	22	6	28	6	30	6	28	8	36	6	28	6	30			0000 1000 00EE EEEE		
	L	**bit# = Dn**	**d =**	2	<10																							0000 rrr1 00EE EEEE		
		bit# = Imm	**d =**	4	<18																							0000 1000 00EE EEEE		
CHK	W	d = Dn	s(bound) =	2	<68	trap		2	<76	2	<76	2	<78	4	<84	4	<86	4	<84	6	<92	4	<84	4	<86	4	<76	0100 DDD1 10ee eeee	If Dn<0 or Dn>(bound), then trap	- * U U U
				2	14	no trap		2	22	2	22	2	24	4	30	4	32	4	30	6	38	4	30	4	32	4	22			
CLR	**B**		**d =**	2	8			2	16	2	16	2	18	4	24	4	26	4	24	6	32							0100 0010 00EE EEEE	0 → d	- 0 1 0 0
	W		**d =**	2	8			2	24	2	24	2	26	4	32	4	34	4	32	6	40							0100 0010 01EE EEEE		
	L		**d =**	2	10			2	40	2	40	2	42	4	48	4	50	4	48	6	56							0100 0010 10EE EEEE		
CMP	B	d = Dn	s =	2	8			2	12	2	12	2	14	4	20	4	22	4	20	6	28	4	20	4	22	4	16	1011 DDD0 00ee eeee	Dn − s	- * * * *
	W	**d = Dn**	**s =**	2	8	2	10	2	16	2	16	2	18	4	24	4	26	4	24	6	32	4	24	4	26	4	16	1011 DDD0 01ee eeee		
	L	d = Dn	s =	2	10	2	10	2	26	2	26	2	28	4	34	4	36	4	34	6	42	4	34	4	36	6	26	1011 DDD0 10ee eeee		

Mnemonic	Size	Address Mode		Dn		An		(An)		(An)+		−(An)		d_{16}(An)		d_8(An,Xn)		Abs.W		Abs.L		d_{16}(PC)		d_8(PC,Xn)		s = Immed d = SR/CC		Opcode Bit Pattern 1111 11 5432 1098 7654 3210	Boolean	Condition Codes X N Z V C
				#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~			
CMPA	W	d = An	s =	2	10	2	10	2	18	2	18	2	20	4	26	4	28	4	26	6	34	4	26	4	28	4	28	1011 AAA0 11ee eeee	An − s	- * * * *
	L	d = An	s =	2	10	2	10	2	26	2	26	2	28	4	34	4	36	4	34	6	42	4	34	4	36	6	26	1011 AAA1 11ee eeee		
CMPI	B	s = Imm	d =	4	16	CMPA		4	20	4	20	4	22	6	28	6	30	6	28	8	36							0000 1100 00EE EEEE	d − #	- * * * *
	W	s = Imm	d =	4	16	CMPA		4	24	4	24	4	26	6	32	6	34	6	32	8	40							0000 1100 01EE EEEE		
	L	s = Imm	d =	6	26	CMPA		6	40	6	40	6	42	8	48	8	50	8	48	10	56							0000 1100 10EE EEEE		
CMPM	B	s = (An)+	d =							2	16																	1011 RRR1 0000 1rrr	d − s	- * * * *
	W	s = (An)+	d =							2	24																	1011 RRR1 0100 1rrr		
	L	s = (An)+	d =							2	40																	1011 RRR1 1000 1rrr		
DBcc	W	d_{16} = Imm				cc	Counter	Branch																				0101 CCCC 1100 1DDD	If cc true, then NOP else Dn − 1 → Dn, If Dn ≠ −1, then PC + disp → PC	- - - - -
		counter =	4	18	false	≠ −1	yes																							
				20	true	N/A	no																							
				26	false	expired	no																							
DIVS	W	d = Dn	s =	2	<162			2	<170	2	<170	2	<172	4	<178	4	<180	4	<178	6	<186	4	<178	4	<180	4	<170	1000 DDD1 11ee eeee	Dn32/s16 → Dn(r : q)	- * * * 0
DIVU	W	d = Dn	s =	2	<144			2	<152	2	<152	2	<154	4	<160	4	<162	4	<160	6	<168	4	<160	4	<162	4	<152	1000 DDD0 11ee eeee	Dn32/s16 → Dn(r : q)	- * * * 0
EOR	B	s = Dn	d =	2	8			2	16	2	16	2	18	4	24	4	26	4	24	6	32							1011 rrr1 00EE EEEE	d ⊕ Dn → d	- * * 0 0
	W	s = Dn	d =	2	8			2	24	2	24	2	26	4	32	4	34	4	32	6	40							1011 rrr1 01EE EEEE		
	L	s = Dn	d =	2	12			2	40	2	40	2	42	4	48	4	50	4	48	6	56							1011 rrr1 10EE EEEE		
EORI	B	s = Imm	d =	4	16			4	24	4	24	4	26	6	32	6	34	6	32	8	40							0000 1010 00EE EEEE	d ⊕ # → d	- * * 0 0
	W	s = Imm	d =	4	16			4	32	4	32	4	34	6	40	6	42	6	40	8	48							0000 1010 01EE EEEE		
	L	s = Imm	d =	6	28			6	56	6	56	6	58	8	64	8	66	8	64	10	72							0000 1010 10EE EEEE		
EORI CCR	B	s = Imm	d =																							4	32	0000 1010 0011 1100	s ⊕ CCR → CCR	* * * * *
EORI SR	W	s = Imm	d =																							4	32	0000 1010 0111 1100	s ⊕ SR → SR	* * * * *
EXG	L	s = Dn	d =	2	10																							1100 DDD1 0100 0DDD	s ↔ d	- - - - -
		s = An	d =			2	10																					1100 AAA1 0100 1AAA		
				2	10	2	10																					1100 DDD1 1000 1AAA		
EXT	W		d =	2	8																							0100 1000 1000 0DDD	bit 7 → bits 15 : 8	- * * 0 0
	L		d =	2	8																							0100 1000 1100 0DDD	bit 15 → bits 31 : 16	
ILLEGAL				2	62																							0100 1010 1111 1100	PC → −(SSP), SR → −(SSP), (illegal vector) → PC	- - - - -
JMP			d =					2	16					4	18	4	22	4	18	6	24	4	18	4	22			0100 1110 11EE EEEE	d → PC	- - - - -
JSR			d =					2	32					4	34	4	38	4	34	6	40	4	34	4	38			0100 1110 10EE EEEE	PC → −(SP), d → PC	- - - - -
LEA	L	d = An	s =																									0100 AAA1 11ee eeee	s → An	- - - - -
LINK		d_{16} = Imm	s =			4	32																					0100 1110 0101 0AAA	An → −(SP), SP → An, SP + disp → SP	- - - - -
LSL, LSR	B/W	count = Dn	d =	2	10 + 2n																							1110 rrrf SS10 1DDD		* * * 0 *
		count = #1–8	d =	2	10 + 2n																							1110 QQQf SS00 1DDD		
	L	count = Dn	d =	2	12 + 2n																							1110 rrrf 1010 1DDD		
		count = #1–8	d =	2	12 + 2n																							1110 QQQf 1000 1DDD		
	Memory W	count = 1	d =					2*	24	2*	24	2*	26	4*	32	4*	34	4*	32	6*	40							1110 001f 11EE EEEE		
MOVE	B	d = Dn	s =	2	8	2	8	2	12	2	12	2	14	4	20	4	22	4	20	6	28	4	20	4	22	4	16	0001 RRRM MMee eeee	s → d	- * * 0 0
		d = An	s =	MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA				
		d = (An)	s =	2	12	2	12	2	16	2	16	2	18	4	24	4	26	4	24	6	32	4	24	4	26	4	20			
		d = (An)+	s =	2	12	2	12	2	16	2	16	2	18	4	24	4	26	4	24	4	32	4	24	4	26	4	20			
		d = −(An)	s =	2	12	2	12	2	16	2	16	2	18	4	24	4	26	4	24	4	32	4	24	4	26	4	20			
		d = d_{16}(An)	s =	4	20	4	20	4	24	4	24	4	26	6	32	6	34	6	32	8	40	6	32	6	34	6	28			
		d = d_8(An,Xn)	s =	4	22	4	22	4	26	4	26	4	28	6	34	6	36	6	34	8	42	6	34	6	36	6	30			
		d = Abs.W	s =	4	20	4	20	4	24	4	24	4	26	6	32	6	34	6	32	8	40	6	32	6	34	6	28			
		d = Abs.L	s =	6	28	6	28	6	32	4	32	6	34	8	40	8	42	8	40	10	48	8	40	8	42	8	36			
	W	d = Dn	s =	2	8	2	8	2	16	2	16	2	18	4	24	4	26	4	24	6	32	4	24	4	26	4	16	0011 RRRM MMee eeee	s → d	- * * 0 0
		d = An	s =	MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA				
		d = (An)	s =	2	16	2	16	2	24	2	24	2	26	4	32	4	34	4	32	6	40	4	32	4	34	4	24			
		d = (An)+	s =	2	16	2	16	2	24	2	24	2	26	4	32	4	34	4	32	4	40	4	32	4	34	4	24			
		d = −(An)	s =	2	16	2	16	2	24	2	24	2	26	4	32	4	34	4	32	6	40	4	32	4	34	4	24			
		d = d_{16}(An)	s =	4	24	4	24	4	32	4	32	4	34	6	40	6	42	6	40	8	48	6	40	6	42	6	32			
		d = d_8(An,Xn)	s =	4	26	4	26	4	34	4	34	4	32	6	42	6	44	6	42	8	50	6	42	6	44	6	34			
		d = Abs.W	s =	4	20	4	20	4	32	4	32	4	34	6	40	6	42	6	40	8	48	6	40	6	42	6	32			
		d = Abs.L	s =	6	32	6	32	6	40	6	40	6	42	8	48	8	50	8	48	10	56	8	48	8	50	8	40			
	L	d = Dn	s =	2	8	2	8	2	24	2	24	2	26	4	32	4	34	4	32	6	40	4	32	4	34	6	24	0010 RRRM MMee eeee	s → d	- * * 0 0
		d = An	s =	MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA		MOVEA				
		d = (An)	s =	2	24	2	24	2	40	2	40	2	42	4	48	4	50	4	48	6	56	4	48	4	50	6	40			
		d = (An)+	s =	2	24	2	24	2	40	2	40	2	42	4	48	4	50	4	48	4	56	4	48	4	50	6	40			
		d = −(An)	s =	2	24	2	24	2	40	2	40	2	42	4	48	4	50	4	48	6	56	4	48	4	50	6	40			
		d = d_{16}(An)	s =	4	32	4	32	4	48	4	48	4	50	6	56	6	58	6	56	8	64	6	56	6	58	8	48			
		d = d_8(An,Xn)	s =	4	34	4	34	4	50	4	50	4	52	6	58	6	60	6	58	8	66	6	58	6	60	8	50			
		d = Abs.W	s =	4	32	4	32	4	48	4	48	4	50	6	56	6	58	6	56	8	64	6	56	6	58	8	48			
		d = Abs.L	s =	6	40	6	40	6	56	6	56	6	58	8	64	8	66	8	64	10	72	8	64	8	66	10	56			
MOVE CCR	W	d = CCR	s =	2	18					2	26	2	26	2	28	4	34	4	36	4	34	6	42	4	34	4	36	4	26	0100 0100 11ee eeee	s → CCR	* * * * *
MOVE SR	W	d = SR	s =	2	18					2	26	2	26	2	28	4	34	4	36	4	34	6	42	4	34	4	36	4	26	0100 0110 11ee eeee	s → SR	* * * * *
		s = SR	d =	2	10					2	24	2	24	2	26	4	32	4	34	4	32	6	40							0100 0000 11EE EEEE	SR → d	- - - - -

Mnemonic	Size	Address Mode		Dn		An		(An)		(An)+		−(An)		d_{16}(An)		dg(An,Xn)		Abs.W		Abs.L		d_{16}(PC)		dg(PC,Xn)		s = Immed d = SR/CC		Opcode Bit Pattern 1111 11 5432 1098 7654 3210	Boolean	Condition Codes X N Z V C
MOVE USP	L	s = USP	d =			2	8																					0100 1110 0110 1AAA	USP → An	- - - - -
		d = USP	s =			2	8																					0100 1110 0110 0AAA	An → USP	
MOVEA	W	d = An	s =	2	8	2	8	2	16	2	16	2	18	4	24	4	26	4	24	6	32	4	24	4	26	4	16	0011 AAA0 01ee eeee	s → An	- - - - -
	L	d = An	s =	2	8	2	8	2	24	2	24	2	26	4	32	4	34	4	32	6	40	4	32	4	34	6	24	0010 AAA0 01ee eeee		
MOVEM	W	s = Xn	d =					4	16 + 8n			4	16 + 8n	6	24 + 8n	6	26 + 8n	6	24 + 8n	8	32 + 8n							0100 1000 10EE EEEE a7 a0 d7 d0	Xn → d	- - - - -
		d = Xn	s =					4	24 + 8n	4	24 + 8n			6	32 + 8n	6	34 + 8n	6	32 + 8n	8	40 + 8n	6	32 + 8n	6	34 + 8n			0100 1100 10ee eeee a7 a0 d7 d0	s → Xn	
	L	s = Xn	d =					4	16 + 16n			4	16 + 16n	6	24 + 16n	6	26 + 16n	6	24 + 16n	8	32 + 16n							0100 1000 11EE EEEE a7 a0 d7 d0	Xn → d	
		d = Xn	s =					4	24 + 16n	4	24 + 16n			6	32 + 16n	6	34 + 16n	6	32 + 16n	8	40 + 16n	6	32 + 16n	6	34 + 16n			0100 1100 11ee eeee a7 a0 d7 d0	s → Xn	
MOVEP	W	s = Dn	d =											4	24													0000 DDD1 1000 1AAA	Dn → d by bytes	- - - - -
		s = d_{16}(An)	d =	4	24																							0000 DDD1 0000 1AAA	s → Dn by bytes	
	L	s = Dn	d =											4	32													0000 DDD1 1100 1AAA	Dn → d by bytes	
		s = d_{16}(An)	d =	4	32																							0000 DDD1 0100 1AAA	s → Dn by bytes	
MOVEQ	L	s = Imm8	d =	2	8																							0111 DDD0 QQQQ QQQQ	# → Dn	- * * 0 0
MULS	W	d = Dn	s =	2	<74			2	<82	2	<82	2	<84	4	<90	4	<92	4	<90	6	<98	4	<90	4	<92	4	<82	1100 DDD1 11ee eeee	Dn × s → Dn	- * * 0 0
MULU	W	d = Dn	s =	2	<74			2	<82	2	<82	2	<84	4	<90	4	<92	4	<90	6	<98	4	<90	4	<92	4	<82	1100 DDD0 11ee eeee	Dn × s → Dn	- * * 0 0
NBCD	B		d =	2	10			2	16	2	16	2	18	4	24	4	26	4	24	6	32							0100 1000 00EE EEEE	0 − d10 − X → d	* U * U *
NEG	B		d =	2	8			2	16	2	16	2	18	4	24	4	26	4	24	6	32							0100 0100 00EE EEEE	0 − d → d	* * * * *
	W		d =	2	8			2	24	2	24	2	26	4	32	4	34	4	32	6	40							0100 0100 01EE EEEE		
	L		d =	2	10			2	40	2	40	2	42	4	48	4	50	4	48	6	56							0100 0100 10EE EEEE		
NEGX	B		d =	2	8			2	16	2	16	2	18	4	24	4	26	4	24	6	32							0100 0000 00EE EEEE	0 − d − X → d	* * * * *
	W		d =	2	8			2	24	2	24	2	26	4	32	4	34	4	32	6	40							0100 0000 01EE EEEE		
	L		d =	2	10			2	40	2	40	2	42	4	48	4	50	4	48	6	56							0100 0000 10EE EEEE		
NOP				2	8																							0100 1110 0111 0001	none	- - - - -
NOT	B		d =	2	8			2	16	2	16	2	18	4	24	4	26	4	24	6	32							0100 0110 00EE EEEE	¬d → d	- * * 0 0
	W		d =	2	8			2	24	2	24	2	26	4	32	4	34	4	32	6	40							0100 0110 01EE EEEE		
	L		d =	2	10			2	40	2	40	2	42	4	48	4	50	4	48	6	56							0100 0110 10EE EEEE		
OR	B	s = Dn	d =					2	16	2	16	2	18	4	24	4	26	4	24	6	32							1000 DDD1 00EE EEEE	d<or>Dn → d	- * * 0 0
		d = Dn	s =	2	8			2	12	2	12	2	14	4	20	4	22	4	20	6	28	4	20	4	22	4	16	1000 DDD0 00ee eeee	Dn<or>s → Dn	
	W	s = Dn	d =					2	24	2	24	2	26	4	32	4	34	4	32	6	40							1000 DDD1 01EE EEEE	d<or>Dn → d	
		d = Dn	s =	2	8			2	16	2	16	2	18	4	24	4	26	4	24	6	32	4	24	4	26	4	16	1000 DDD0 01ee eeee	Dn<or>s → Dn	
	L	s = Dn	d =					2	40	2	40	2	42	4	48	4	50	4	48	6	56							1000 DDD1 10EE EEEE	d<or>Dn → d	
		d = Dn	s =	2	12			2	26	2	26	2	28	4	34	4	36	4	34	6	42	4	34	4	36	6	28	1000 DDD0 10ee eeee	Dn<or>s → Dn	
ORI	B	s = Imm	d =	4	16			4	24	4	24	4	26	6	32	6	34	6	32	8	40							0000 0000 00EE EEEE	d<or># → d	- * * 0 0
	W	s = Imm	d =	4	16			4	32	4	32	4	34	6	40	6	42	6	40	8	48							0000 0000 01EE EEEE		
	L	s = Imm	d =	6	28			6	56	6	56	6	58	8	64	8	66	8	64	10	72							0000 0000 10EE EEEE		
ORI CCR	B	s = Imm	d =																							4	32	0000 0000 0011 1100	s<or>CCR → CCR	* * * * *
ORI SR	W	s = Imm	d =																							4	32	0000 0000 0111 1100	s<or>SR → SR	* * * * *
PEA	L		s =					2	24					4	32	4	36	4	32	6	40	4	32	4	36			0100 1000 01ee eeee	s → −(SP)	- - - - -
RESET				2	136																							0100 1110 0111 0000	assert RESET pin	- - - - -
ROL, ROR	B/W	count = Dn	d =	2	10 + 2n																							1110 rrrf SS11 1DDD		- * * 0 0
		count = #1–8	d =	2	10 + 2n																							1110 QQQf SS01 1DDD		
	L	count = Dn	d =	2	12 + 2n																							1110 rrrf 1011 1DDD		
		count = #1–8	d =	2	12 + 2n																							1110 QQQf 1001 1DDD		
	Memory	count = 1	d =					2*	32	2*	32	2*	34	4*	40	4*	42	4*	40	6*	48							1100 011f 11EE EEEE		
ROXL, ROXR	B/W	count = Dn	d =	2	10 + 2n																							1110 rrrf SS11 0DDD		* * * 0 *
		count = #1–8	d =	2	10 + 2n																							1110 QQQf SS01 0DDD		
	L	count = Dn	d =	2	12 + 2n																							1110 rrrf 1011 0DDD		
		count = #1–8	d =	2	12 + 2n																							1110 QQQf 1001 0DDD		
	Memory	count = 1	d =					2*	32	2*	32	2*	34	4*	40	4*	42	4*	40	6*	48							1110 010f 11EE EEEE		
RTE				2	40																							0100 1110 0111 0011	(SP) + → SR, (SP) + → PC	* * * * *
RTR				2	40																							0100 1110 0111 0111	(SP) + → CC, (SP) + → PC	* * * * *
RTS				2	32																							0100 1110 0111 0101	(SP) + → PC	- - - - -
SBCD	B	s = Dn	d =	2	10																							1000 RRR1 0000 0rrr	d10 − s10 − X → d	* U * U *
		s = −(An)	d =									2	20															1000 RRR1 0000 1rrr		
Scc	B	cc = True	d =	2	10			2	16	2	16	2	18	4	24	4	26	6	24	6	32							0101 CCCC 11EE EEEE	If cc true, then 1s → d	- - - - -
		cc = False	d =	2	8			2	16	2	16	2	18	4	24	4	26	6	24	6	32								else 0s → d	
STOP																										4	4	0100 1110 0111 0010	# → SR, wait for interrupt	* * * * *
SUB	B	s = Dn	d =				SUBA	2	16	2	16	2	18	4	24	4	26	4	24	6	32							1001 DDD1 00EE EEEE	d − Dn → d	* * * * *
		d = Dn	s =	2	8	2	8	2	12	2	12	2	14	4	20	4	22	4	20	6	28	4	20	4	22	4	16	1001 DDD0 00ee eeee	Dn − s → Dn	
	W	s = Dn	d =				SUBA	2	24	2	24	2	26	4	32	4	34	4	32	6	40							1001 DDD1 01EE EEEE	d − Dn → d	
		d = Dn	s =	2	8	2	8	2	16	2	16	2	18	4	24	4	26	4	24	6	32	4	24	4	26	4	16	1001 DDD0 01ee eeee	Dn − s → Dn	
	L	s = Dn	d =				SUBA	2	40	2	40	2	42	4	48	4	50	4	48	6	56							1001 DDD1 10EE EEEE	d − Dn → d	
		d = Dn	s =	2	12	2	12	2	26	2	26	2	28	4	34	4	36	4	34	6	42	4	34	4	36	6	28	1001 DDD0 10ee eeee	Dn − s → Dn	

Mnemonic	Size	Address Mode		Dn #	Dn ~	An #	An ~	(An) #	(An) ~	(An)+ #	(An)+ ~	−(An) #	−(An) ~	d_{16}(An) #	d_{16}(An) ~	d_8(An,Xn) #	d_8(An,Xn) ~	Abs.W #	Abs.W ~	Abs.L #	Abs.L ~	d_{16}(PC) #	d_{16}(PC) ~	d_8(PC,Xn) #	d_8(PC,Xn) ~	s = Immed d = SR/CC #	s = Immed d = SR/CC ~	Opcode Bit Pattern 1111 11 5432 1098 7654 3210	Boolean	Condition Codes X N Z V C
SUBA	W	d = An	s =	2	12	2	12	2	20	2	20	2	22	4	28	4	30	4	28	6	36	4	28	4	30	4	20	1001 AAA0 11ee eeee	An − s → An	- - - - -
	L	d = An	s =	2	12	2	12	2	26	2	26	2	28	4	34	4	36	4	34	6	42	4	34	4	36	6	28	1001 AAA1 11ee eeee		
SUBI	B	s = Imm	d =	4	16	SUBA		4	16	4	16	4	18	6	24	6	26	6	24	8	32							0000 0100 00EE EEEE	d − # → d	* * * * *
	W	s = Imm	d =	6	16	SUBA		4	24	4	24	4	26	6	32	6	34	6	32	8	40							0000 0100 01EE EEEE		
	L	s = Imm	d =	6	28	SUBA		6	40	6	40	6	42	8	48	8	50	8	48	10	56							0000 0100 10EE EEEE		
SUBQ	B	s = Imm3	d =	2	8			2	24	2	24	2	26	4	32	4	34	4	32	6	40							0101 QQQ1 00EE EEEE	d − # → d	* * * * *
	W	s = Imm3	d =	2	8	2	12	2	32	2	32	2	34	4	40	4	42	4	40	6	48							0101 QQQ1 01EE EEEE		
	L	s = Imm3	d =	2	12	2	12	2	56	2	56	2	58	4	64	4	66	4	64	6	72							0101 QQQ1 10EE EEEE		
SUBX	B	s = Dn	d =	2	8																							1001 RRR1 0000 0rrr	d − s − X → d	* * * * *
		s = −(An)	d =									2	22															1001 RRR1 0000 1rrr		
	W	s = Dn	d =	2	8																							1001 RRR1 0100 0rrr		
		s = −(An)	d =									2	50															1001 RRR1 0100 1rrr		
	L	s = Dn	d =	2	12																							1001 RRR1 1000 0rrr		
		s = −(An)	d =									2	58															1001 RRR1 1000 1rrr		
SWAP	W		d =	2	8																							0100 1000 0100 0DDD	Dn(31 : 16) ⟷ Dn(15 : 0)	- * * 0 0
TAS	B		d =	2	8			2	18	2	18	2	20	4	26	4	28	4	26	6	34							0100 1010 11EE EEEE	test d → cc, 1 → bit 7 of d	- * * 0 0
TRAP				2	62																							0100 1110 0100 VVVV	PC → −(SSP), SR → −(SSP), (1 of 16 trap vectors) → PC	- - - - -
TRAPV				2	66	trap taken																						0100 1110 0111 0110	If V = 1, then PC → −(SSP), SR → −(SSP), (TRAPV vector) → PC, else NOP	- - - - -
				2	8	trap not taken																								
TST	B		d =	2	8			2	12	2	12	2	14	4	20	4	22	4	20	6	28							0100 1010 00EE EEEE	test d → cc	- * * 0 0
	W		d =	2	8			2	16	2	16	2	18	4	24	4	26	4	24	6	32							0100 1010 01EE EEEE		
	L		d =	2	8			2	24	2	24	2	26	4	32	4	34	4	32	6	40							0100 1010 10EE EEEE		
UNLK						2	24																					0100 1110 0101 1AAA	An → SP, (SP) + → An	- - - - -

MC68008/MC68010/MC68012 Programming Reference Card Timing Table Notes

General Notes:

*	Word Only	'	Complement (invert)
#	Number of Bytes in Instruction	d_8	8-Bit Displacement
~	Execution Time in Clock Periods	d_{16}	16-Bit Displacement
s	Source (s10 = base 10 operand)	Imm	Immediate Data
d	Destination (d10 = base 10 operand)	Imm3	Immediate Data, 3 Bits
<	Value is Maximum Number	Imm8	Immediate Data, 8 Bits

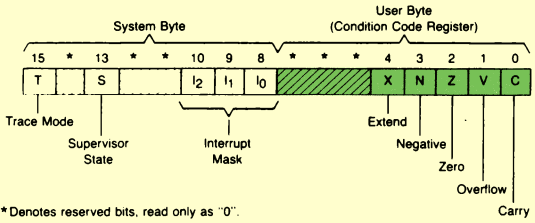
Opcode Bit Pattern Codes:

A	Address Register Number	f	Direction: 0 = Right 1 = Left	R	Destination Register
C	Test Condition			r	Source Register
D	Data Register Number	M	Destination EA Mode	S	Size: 00 = Byte 01 = Word 10 = Long
E	Destination Effective Address	P	Displacement		
e	Source Effective Address	Q	Quick Immediate Data		
V	Vector Number				
XX	Move size: 01 = Byte 11 = Word				

Condition Code Notation:

*	Set according to result of operation.
—	Not affected by operation.
0	Cleared
1	Set
U	Undefined after operation.

Status Register



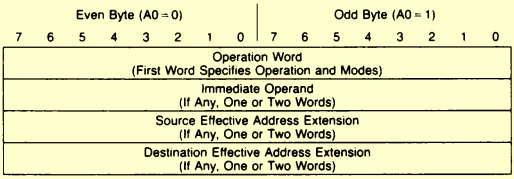
*Denotes reserved bits, read only as "0".

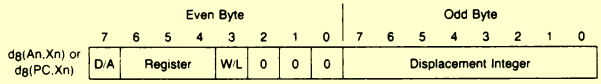
Interrupt Encoding

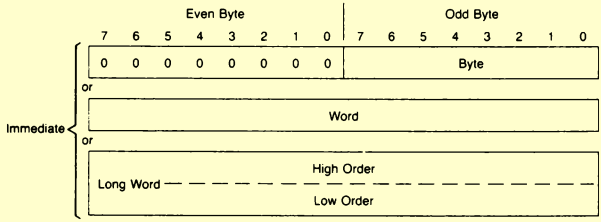
Priority	$\overline{IPL2}/\overline{1}/\overline{0}$ Control Lines	Requested Interrupt Level	Status Reg. Int. Mask I2/I1/I0	Recognized Interrupt Level
Highest	LLL	7*	111	7
•	LLH	6	110	7
•	LHL	5*	101	6,7
•	LHH	4	100	5-7
•	HLL	3	011	4-7
•	HLH	2*	010	3-7
•	HHL	1	001	2-7
Lowest	HHH	None*	000	1-7

Note: Asterisk denotes supported interrupt levels for 48-pin device. All interrupt levels are supported for 52-pin device.

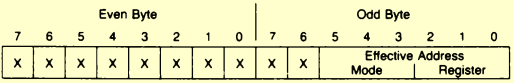
Instruction Operation Word General Format


Even Byte (A0 = 0)
Odd Byte (A0 = 1)
7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0
Operation Word
(First Word Specifies Operation and Modes)
Immediate Operand
(If Any, One or Two Words)
Source Effective Address Extension
(If Any, One or Two Words)
Destination Effective Address Extension
(If Any, One or Two Words)

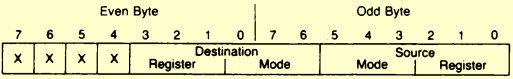




Single-Effective-Address Instruction Operation Word



Double-Effective-Address Instruction Operation Word

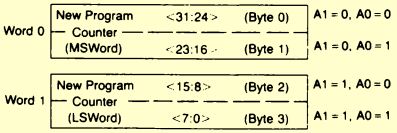


Reference Classification

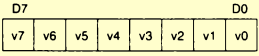
Function Code Output			
FC2	FC1	FC0	Reference Class
0	0	0	(Unassigned)
0	0	1	User Data
0	1	0	User Program
0	1	1	(Unassigned)

Function Code Output			
FC2	FC1	FC0	Reference Class
1	0	0	(Unassigned)
1	0	1	Supervisor Data
1	1	0	Supervisor Program
1	1	1	Interrupt Acknowledge

Exception Vector Format

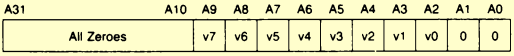


Peripheral Vector Number Format

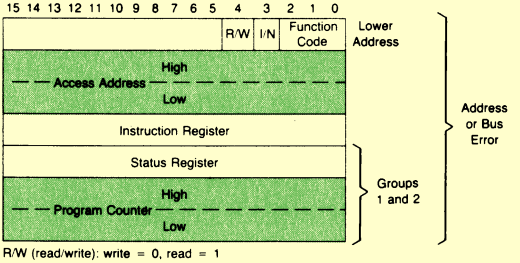


Where: v7 is the MSB of the Vector Number
v0 is the LSB of the Vector Number

Address Translated from 8-Bit Vector Number



Supervisor Stack Order for Bus or Address Error Exception



R/W (read/write): write = 0, read = 1
I/N (instruction/not): instruction = 0, not = 1

Exception Grouping and Priority

Group	Exception	Processing
0	Reset Address Error Bus Error	Exception processing begins within two clock cycles.
1	Trace Interrupt Illegal Privilege	Exception processing begins before the next instruction.
2	TRAP, TRAPV, CHK Zero Divide	Exception processing is started by normal instruction execution.

Exception Vector Assignment

Vector Number(s)	Address			Assignment
	Dec	Hex	Space[6]	
0	0	000	SP	Reset: Initial SSP[2]
1	4	004	SP	Reset: Initial PC[2]
2	8	008	SD	Bus Error
3	12	00C	SD	Address Error
4	16	010	SD	Illegal Instruction
5	20	014	SD	Zero Divide
6	24	018	SD	CHK Instruction
7	28	01C	SD	TRAPV Instruction
8	32	020	SD	Privilege Violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 Emulator
11	44	02C	SD	Line 1111 Emulator
12[1]	48	030	SD	(Unassigned, Reserved)
13[1]	52	034	SD	(Unassigned, Reserved)
14	56	038	SD	Format Error[5]
15	60	03C	SD	Uninitialized Interrupt Vector
16-23[1]	64	040	SD	(Unassigned, Reserved)
	95	05F		—
24	96	060	SD	Spurious Interrupt[3]
25	100	064	SD	Level 1 Interrupt Autovector
26	104	068	SD	Level 2 Interrupt Autovector
27	108	06C	SD	Level 3 Interrupt Autovector
28	112	070	SD	Level 4 Interrupt Autovector
29	116	074	SD	Level 5 Interrupt Autovector
30	120	078	SD	Level 6 Interrupt Autovector
31	124	07C	SD	Level 7 Interrupt Autovector
32-47	128	080	SD	TRAP Instruction Vectors[4]
	191	0BF		
48-63[1]	192	0C0	SD	(Unassigned, Reserved)
	255	0FF		—
64-255	256	100	SD	User Interrupt Vectors
	1023	3FF		—

NOTES:

1. Vector numbers 12, 13, 16 through 23, and 48 through 63 are reserved for future enhancements by Motorola. No user peripheral devices should be assigned these numbers.
2. Reset vector (0) requires four words, unlike the other vectors which only require two words, and is located in the supervisor program space.
3. The spurious interrupt vector is taken when there is a bus error indication during interrupt processing.
4. Trap #n uses vector number 32 + n.
5. MC68010 MC68012 only.
   This vector is unassigned, reserved on the MC68000, and MC68008.
6. SP denotes supervisor program space, and SD denotes supervisor data space.

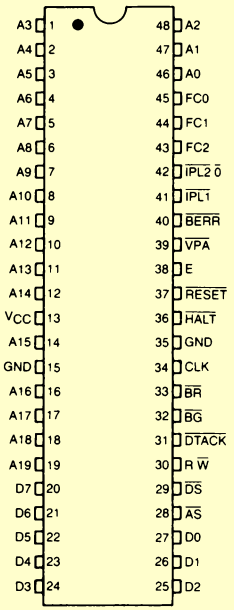
Exception Processing Execution Times

Exception	Periods
Address Error	94(8 14)
Bus Error	94(8 14)
CHK Instruction	68(8 6) +
Divide by Zero	66(8 6) +
Illegal Instruction	62(8/6)
Interrupt	72(9/6)*
Privilege Violation	62(8/6)
RESET**	64(12/0)
Trace	62(8 6)
TRAP Instruction	62(8 6)
TRAPV Instruction	66(10 6)

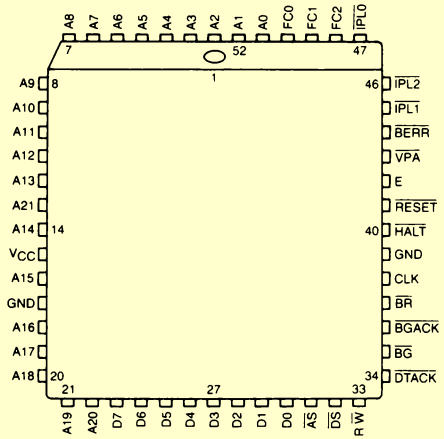
+ Add effective address calculation time.
*The interrupt acknowledge cycle is assumed to take four clock periods.
**Indicates the time from when $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ are first sampled as negated to when instruction execution starts.

Pin Assignments

48-Pin Dual-in-Line Package



52-Lead Quad Package



Powers of 16, Powers of 2

16^m m=	2^n n=	Value
0	0	1
	1	2
	2	4
	3	8
1	4	16
	5	32
	6	64
	7	128
2	8	256
	9	512
	10	1,024
	11	2,048
3	12	4,096
	13	8,192
	14	16,384
	15	32,768

16^m m=	2^n n=	Value
4	16	65,536
	17	131,072
	18	262,144
	19	524,288
5	20	1,048,576
	21	2,097,152
	22	4,194,304
	23	8,388,608
6	24	16,777,216
	25	33,554,432
	26	67,108,864
	27	134,217,728
7	28	268,435,456
	29	536,870,912
	30	1,073,741,824
	31	2,147,483,648
8	32	4,294,967,296

ASCII Character Set (7-Bit Code)

LS Dig. \ MS Dig.	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

Hexadecimal and Decimal Conversion

How to use:

Conversion to Decimal: Find the decimal weights for corresponding hexadecimal characters beginning with the least significant character. The sum of the decimal weights is the decimal value of the hexadecimal number.

Conversion to Hexadecimal: Find the highest decimal value in the table which is lower than or equal to the decimal number to be converted. The corresponding hexadecimal character is the most significant. Subtract the decimal value found from the decimal number to be converted. With the difference repeat the process to find subsequent hexadecimal characters.

23 Byte 16				15 Byte 8				7 Byte 0			
23 Char 20		19 Char 16		15 Char 12		11 Char 8		7 Char 4		3 Char 0	
Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec
0	0	0	0	0	0	0	0	0	0	0	0
1	1,048,576	1	65,536	1	4,096	1	256	1	16	1	1
2	2,097,158	2	131,072	2	8,192	2	512	2	32	2	2
3	3,145,728	3	196,608	3	12,288	3	768	3	48	3	3
4	4,194,304	4	262,144	4	16,384	4	1024	4	64	4	4
5	5,242,880	5	327,680	5	20,480	5	1280	5	80	5	5
6	6,291,456	6	393,216	6	24,576	6	1536	6	96	6	6
7	7,340,032	7	458,752	7	28,672	7	1792	7	112	7	7
8	8,388,608	8	524,288	8	32,768	8	2048	8	128	8	8
9	9,437,184	9	589,824	9	36,864	9	2304	9	144	9	9
A	10,485,760	A	655,360	A	40,960	A	2560	A	160	A	10
B	11,534,336	B	720,896	B	45,056	B	2816	B	176	B	11
C	12,582,912	C	786,432	C	49,152	C	3072	C	192	C	12
D	13,631,488	D	851,968	D	53,248	D	3328	D	208	D	13
E	14,680,064	E	917,504	E	57,344	E	3584	E	224	E	14
F	15,728,640	F	983,040	F	61,440	F	3840	F	240	F	15

Motorola reserves the right to make changes to this product. Although this information has been carefully reviewed and is believed to be reliable, Motorola does not assume any liability arising out of its use.

Printed in England by The Campfield Press 6/85

This was brought to you

from the archives of

http://retro-commodore.eu